IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Applicant:	Spracklen et al.	
App. No.:	10/822,390	Con. No.: 2873
Filed:	April 12, 2004	Art Unit: 2181
Title:	EXECUTION DISPLACEMENT READ-WRITE ALIAS PREDICTION	Examiner: Moll, Jesse R.

RESPONSE TO NON-COMPLIANT APPEAL BRIEF AND APPEAL BRIEF IN SUPPORT OF APPELLANT'S APPEAL TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicant (hereafter "Appellant") hereby submits this Appeal Brief in response to the to Notice of Non-Compliant Appeal Brief mailed August 21, 2007 in the above-captioned case. Appellant respectfully requests consideration of this appeal by the Board of Patent Appeals and Interferences (hereafter the "Board") for allowance of the above-captioned patent application.

An oral hearing is not requested at this time.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The invention is assigned to Sun Microsystems, Inc. of 4150 Network Circle, Santa Clara, California 95054.

II. RELATED APPEALS AND INTERFERENCES

To the best of Appellant's knowledge, there are no appeals or interferences that are related to, will directly affect, will be directly affected by, or have a bearing on the Board's decision in the present appeal.

III. STATUS OF THE CLAIMS

Claims 1-66 are currently pending in the above-referenced application. Claims 58-61 are allowed and claims 4-9, 65 and 66 would be allowable if rewritten in independent form.

The rejection of claims 1-57 and 62-66 is being appealed.

IV. STATUS OF AMENDMENTS

Claims 16-47 were finally rejected under 35 U.S.C. § 101 and claims 1-3, 10-47, 54-57 and 62-64 were finally rejected under 35 U.S.C. § 102(a) in the final Office action mailed December 5, 2006. In response to the final Office action mailed December 5, 2006, Appellant filed a response after final pursuant to 37 C.F.R. § 1.116 on February 5, 2007, amending claims 16, 23, 25, 28, 30 and 33, canceling claims 22 and 29 with claims 1-21, 23-28 and 30-66 remaining pending. The proposed amendments to claims 16, 23, 25, 28, 30 and 33 were done to overcome the 35 U.S.C. § 101 rejection of claims 16-47.

Subsequently, an Advisory action was mailed on February 21, 2007 denying entry of the amendment after final stating that the claim amendments change the scope of multiple claims and maintaining all rejections under 35 U.S.C. § 101 and §102(a) from the final Office action.

However, we fail to understand how the scope of the claims were changed when all that was amended was revision of independent claims 16 and 28 to include the subject matter of dependent claims 22 and 29, respectively, in order to clearly overcome the § 101 concerns, and thus remove those issues for appeal. A copy of all claims on appeal is attached hereto as the Appendix of Claims. Appellant respectfully traverses the 35 U.S.C. § 101 and § 102(a) grounds of rejection.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Independent claims 1, 16, 28, 38, 48, 54 and 62 are similar in inventive scope. Independent claim 1 discloses a processor, independent claims 16, 28 and 38 disclose methods, independent claim 48 discloses a computer program product and independent claims 54 and 62 disclose apparatus. Each of these independent claims is generally directed toward predicting aliasing between read type instructions and write type instructions that reference the same memory locations. The alias prediction is based on a displacement between the instructions and on the previous detection of aliasing of the two instructions.

1. Claim 1

Claim 1 discloses a processor that predicts aliasing between read type and write type instructions. See *specification, paragraph 1024.* Aliasing may occur when a load operation accesses a memory location that has been modified by a store operation, resulting in a data hazard. See *specification, paragraph 1002.* A read type instruction may be any type of processor/register load instruction. See *specification, paragraph 1039.* A write type instruction may be any type of processor/register store instruction. See *specification, paragraph 1034.* The alias prediction is based at least in part on respective displacements between the read type and write type instructions. See *specification, paragraphs 1025-6.* The displacement is an execution displacement that measures the displacement of the aliasing operations based on the difference or distance between operations with respect to program execution or operation sequence execution. That is, execution displacement can track aliasing between operations in different loop iterations. See *specification, paragraph 1025.*

Alias prediction is also based on the previous detection of respective aliasings between the read type instructions and the write type instructions. See *specification, paragraphs 1052-5.* Static (e.g., program counter) and dynamic (e.g., rename identifier) identifiers are used to determine aliasing operations. See *specification, paragraph 1028.* Data is bypassed from the write type instructions to the corresponding predicted to alias read type instructions using register information of the aliasing write type instructions. See *specification, paragraph 1048.*

2. Claim 16

Claim 16 discloses a method for predicting aliasing between read type and write type instructions. See *Specification, paragraph 1028 and Figure 2.* A register rename stage is used to track a write type instruction and a read type instruction which have been previously indicated as aliased. The register rename stage may be an operation rename unit 101 that manages renaming of architectural registers to working registers. See *Specification, paragraphs 1025 and 1034.* The operation rename unit may use aliased read op encoding

105 and aliased write op encoding **111** to track aliasing. See *Specification, paragraph 1024.* A read type instruction may be any type of processor/register load instruction. See *specification, paragraph 1039.* A write type instruction may be any type of processor/register store instruction. See *specification, paragraph 1034.* The previous indication of aliasing between an instance of the write type instruction and an instance of the read type instruction may be identified in the aliased read operation encoding. See *Specification, paragraph 1024.* For example, a loop may have a read and a write type instruction that may result in the read type instruction of one loop iteration aliasing with the write type instruction of another loop iteration. Aliasing may occur when a load operation accesses a memory location that has been modified by a store operation, resulting in a data hazard. See *specification, paragraph 1002.*

The method further comprises predicting a current instance of the read type instruction will alias with a current instance of the write type instruction when the displacement between the two instructions matches the displacement between the previous indication of aliasing. See *Specification, paragraphs 1028-9.* The displacement may be execution displacement (so that a store in a first loop iteration that aliases with a read in a second loop iteration can be detected). Static identifiers are used to track the instances. See *Specification, paragraphs 1029, 1052.* A static identifier may be a program counter. See *specification, paragraph 1028.*

<u>3. Claim 28</u>

Claim 28 discloses a method for predicting aliasing between read type and write type instructions. The method involves observing a repeated pattern of aliasing between a write type instruction and a read type instruction based on static identifiers of the instructions. See *Specification, paragraphs 1052-3.* The static identifiers may be the program counter values. See *Specification 1028.* That is, the program is executing a loop such that the same load and store instruction are repetitively executed. The program counter values for the load and stores remain the same for each loop iteration.

The method further involves determining a displacement between the aliasing instances of the write type instruction and the read type instruction based on dynamic identifiers. See *Specification, paragraph 1052-3.* The dynamic identifier may be the rename identifier. See *Specification, paragraph 1028.* For example, the store instruction in a first loop iteration may have a dynamic identifier of 55 and this store aliases with a read instruction in a third iteration of the loop having a dynamic identifier of 67, resulting in a dynamic displacement of 12 between the aliasing instances of the write and read instructions.

The method finally involves predicting aliasing between a current instance of the read instruction identified by its static identifier and a store instruction having a dynamic identifier

that is located using the dynamic identifier of the read instruction and the displacement. *See Specification, paragraph 1052.* For example, in a repeating loop, a read instruction may have a static identifier of 155 (its program counter value). This load has been previously indicated as aliasing with a store instruction with a displacement of 12. During the next loop iteration, the load operation static identifier remains at 155. The static identifier thus identifies the dynamic identifier of the load instruction for the current loop. The dynamic identifier of the load and the displacement can be used to determine the dynamic identifier of the store instruction predicted to alias with the load instruction.

4. Claim 38

Claim 38 discloses a method for predicting aliasing between a read and a write instruction and bypassing data from the write instruction to the load instruction. The method involves detecting aliasing between a read type instruction and a write type instruction. Aliasing may occur when a load operation accesses a memory location that has been modified by a store operation, resulting in a data hazard. *See specification, paragraph 1002.* A read type instruction may be any type of processor/register load instruction. *See specification, paragraph 1039.* A write type instruction may be any type of processor/register store instruction. *See specification, paragraph 1034.*

The method further involves determining displacement between the read type instruction and the write type instruction, wherein the displacement is with respect to program execution. Displacement with respect to program execution is execution displacement. *See Specification, paragraph 1025.* Execution displacement is calculated using dynamic identifiers. *See Specification, paragraph 1029.*

The method further involves observing repeated aliasing between subsequent instances of the read type instruction and the write type instruction. *See Specification, paragraphs 1028-9.* The method then selects a current instance of the write type instruction based on the displacement and a current instance of the read type instruction. *See Specification, Figure 4B, operations 415, and paragraph 1044.*

Finally, the method involves bypassing data from a data source of the current write type instruction to a data destination of the current read type instruction. *See Specification, Figure 4B, operation 421, and paragraph 1047-1050.*

5. Claim 48

Claim 48 discloses a computer program product. The computer program product comprises three sequences of instructions. The first sequence of instructions, when executed, update a first encoding with a read type instruction's static identifier and a displacement between an instance of the read type instruction and an instance of a write type instruction observed as aliasing with the read type instruction when the static identifier of the read type instruction is not present in the first encoding. Otherwise the first encoding

is updated to indicate repeated aliasing and a second encoding is updated with the write type instruction's static identifier. *See Specification, paragraphs 1028-9; Figure 2, blocks 203, 205 and 207.* The first encoding may be the aliased read op encoding 503, the second encoding may be the aliased write op encoding 505 of Figure 5A. A read type instruction may be any type of processor/register load instruction. *See specification, paragraph 1039.* A write type instruction may be any type of processor/register store instruction. *See specification, paragraph 1034.* An instance of a read type instruction or a write type instruction is a particular instance of a read type instruction or write type instruction that has the same static identifier but whose dynamic identity changes over time or with respect to program execution (i.e., a load or store within a loop). *See Specification, paragraph 1028.* The displacement is an execution displacement that measures the displacement of the aliasing operations based on the difference or distance between operations with respect to program execution or operation sequence execution. That is, execution displacement can track aliasing between operations in different loop iterations. *See specification, paragraph 1025.* The static identifier may be the program counter and the dynamic identifier may be the rename identifier. *See Specification, paragraph 1028.*

The second sequence of instructions, when executed, update a third encoding with a dynamic identifier of an instance of a write type instruction when the static identifier of the write type instruction instance is found in the second encoding. *See Specification, paragraph 1034 and Figure 3.* The third encoding may be the alias register bypass encoding 507 shown in Figure 5A.

The third sequence of instructions, when executed, bypass data from an instance of a write type instruction to an instance of a read type instruction with register information of the write type instruction instance based at least in part on displacement between the instances as indicated by corresponding dynamic identifiers. *See Specification, paragraphs 1047-50 and block 421 of Figure 4B.*

6. Claim 54

Claim 54 discloses an apparatus for predicting aliasing between read type instructions and write type instructions that reference the same memory locations. More specifically, the apparatus includes a data hazard detection module. The data hazard detection module may include a memory disambiguation buffer, load store queue, etc. *See Specification, paragraph 1024 and Figure 1.*

The apparatus further includes a means for predicting aliasing between a current instance of a read type instruction and a current instance of a write type instruction based on displacement between the current instance of the instructions and displacement between previously observed aliased instances of the read type instruction and the write type instruction. Aliasing may occur when a load operation accesses a memory location that has

been modified by a store operation, resulting in a data hazard. See *specification, paragraph 1002*. A read type instruction may be any type of processor/register load instruction. See *specification, paragraph 1039*. A write type instruction may be any type of processor/register store instruction. See *specification, paragraph 1034*. An instance of a read type instruction or a write type instruction is a particular instance of a read type instruction or write type instruction that has the same static identifier but whose dynamic identity changes over time or with respect to program execution (i.e., a load or store within a loop). See *Specification, paragraph 1028*.

The means for predicting aliasing involves an operation rename unit 101 that includes an aliased read operation encoding 105 and an aliased write operation encoding 111. These encodings may be implemented by hardware tables, data structures, a single encoding with multiple access points or plural encodings. The collection of encodings may be referred to as an alias predictor. See *Specification, paragraphs 1024, 1026 and Figure 1*. Figure 5A depicts example encodings: aliased read op encoding 503, aliased write op encoding 505 and alias prediction register bypass encoding 507.

7. Claim 62

Claim 62 discloses an apparatus for predicting aliasing between read type instructions and write type instructions that reference the same memory locations. More specifically, the apparatus includes an alias predictor that includes one or more structures to host indications of write type instructions and particular instances of the write type instructions and read type instructions, respective execution displacements between the instances and register information. The alias predictor predicts aliasings between read and write type instruction instances based on the structures and indications of detected aliasings between the instruction instances. The alias predictor may be a collection of encodings including an aliased read operation encoding 105, an aliased write operation encoding 111 and an alias prediction register bypass encoding 113. These encodings may be implemented by hardware tables, data structures, a single encoding with multiple access points or plural encodings. The collection of encodings may be referred to as an alias predictor. See *Specification, paragraph 1024*.

Aliasing may occur when a load operation accesses a memory location that has been modified by a store operation, resulting in a data hazard. See *specification, paragraph 1002*. A read type instruction may be any type of processor/register load instruction. See *specification, paragraph 1039*. A write type instruction may be any type of processor/register store instruction. See *specification, paragraph 1034*. An instance of a read type instruction or a write type instruction is a particular instance of a read type instruction or write type instruction that has the same static identifier but whose dynamic identity changes over time or with respect to program execution (i.e., a load or store within a loop). See *Specification,*

paragraph 1028. The alias prediction is based at least in part on respective displacements between the read type and write type instructions. *See specification, paragraphs 1025-6.* The displacement is an execution displacement that measures the displacement of the aliasing operations based on the difference or distance between operations with respect to program execution or operation sequence execution. That is, execution displacement can track aliasing between operations in different loop iterations. *See specification, paragraph 1025.*

The apparatus further includes a rename unit coupled with the alias predictor. The rename unit provides register information for write type instruction instances to the alias predictor. The rename unit may be an operation rename unit 101. *See Specification, paragraph 1025 and Figure 1.*

The apparatus finally includes a data hazard detection unit coupled with the alias predictor. The data hazard detection unit detects aliasing between particular instances of read and write type instructions and indicates the detected aliasings to the alias predictor. The data hazard detection module may include a memory disambiguation buffer, load store queue, etc. *See Specification, paragraph 1024 and Figure 1.*

VI. GROUNDS OF REJECTION

I. Whether claims 1-3, 10-47, 54-57 and 62-64 are unpatentable under 35 U.S.C. § 102(a) as being anticipated by Shen et al., "*Modern Processor Design-Fundamentals of Superscalar Processors*," 2003, McGraw-Hill Higher Education, Beta Edition, Chapter 4, pages 196-202 (hereinafter "Shen").

II. Whether claims 22, 29 and 38 are unpatentable under 35 U.S.C. § 101 for failing to produce a useful, tangible and concrete result.

VII. ARGUMENT

I. THE REJECTION OF CLAIMS 1-3, 10-47, 54-57 AND 62-64 UNDER 35 U.S.C. § 102(a) IS IMPROPER BECAUSE THE REFERENCE DOES NOT DISCLOSE ALL THE ELEMENTS OF THE CLAIMS

Claims 1-3, 10-47, 54-57 and 62-64 are rejected under 35 U.S.C. § 102(a) as being anticipated by Shen. In order for a claim to be anticipated under 35 U.S.C. § 102, "each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *See* MPEP § 2131, *see also Verdegaal Bros. v. Union Oil Co. of California*, 841 F.2d 628, 631, USPQ2d 1051, 1053 (Fed. Cir. 1987). For the reasons cited below, Appellant respectfully submits that Shen does not anticipate claims 1-3, 10-47, 54-57 and 62-64.

A. Independent claims 1, 16, 28, 38, 48, 54 and 62 are patentable over Shen

Claims 1, 16, 28, 38, 48, 54 and 62 are independent claims from which the other rejected claims depend. Accordingly, our initial arguments focus on the independent claims.

Appellant submits that Shen does not disclose predicting "aliasing between read type instructions and write type instructions based at least in part on <u>respective displacements</u> between the read type and write type instructions and on <u>previous detection of respective aliasings</u> between the read type instructions and the write type instructions" as recited by claim 1 and similarly recited by claims 16, 28, 38, 48, 54 and 62 (emphasis provided). That is, the alias prediction involves detecting read type instructions and write type instructions that are likely to alias based on (1) the displacement between the instructions (e.g., the number of instructions between them during program execution) and (2) the previous detection of aliasing of the instructions.

The Examiner asserts that Figure 4-43 of Shen teaches alias prediction based on displacement. *See final Office action, page 4.* The Examiner alleges that Figure 4-43 discloses aliasing of load and store instructions that exist in a loop. The Appellant asserts that Figure 4-43 of Shen at most teaches the concept of moving a load ahead of a store (load bypassing) and forwarding store data directly to the load instruction (load forwarding). This has nothing to do with alias prediction based on a displacement. As such, Shen only discloses the use of memory addresses to predict aliasing of store and load instructions.

More specifically, in the context of load bypassing when a load instruction is issued, Shen discloses alias prediction by looking in a store buffer to determine if any of the addresses of previously issued stores still in flight alias with the load instruction address. *See Shen page 198, first full paragraph.* Further, for load forwarding alias prediction, Shen discloses comparing previous store addresses with the load address to determine aliasing and if there are multiple store addresses that alias with the load, then using the most recent

store that aliased with the load. *See Shen page 199, first full paragraph.* Shen does not base alias prediction on displacement between aliasing instructions because Shen only looks at memory addresses and not program counter values or rename identifiers that allow aliasing instructions to be identified based on the number of instructions executed between them as required by the independent claims.

The Appellant further asserts that Shen does not disclose alias prediction based on previous detection of aliasing between instructions as recited by claim 1 and similarly by independent claims 16, 28, 38, 48, 54 and 62. The Examiner alleges that Shen teaches previous detection of respective aliasings by observing repeated aliasing such as the load and store instructions existing in a loop as shown in Figure 4-43. *See final Office action, page 5.* The Applicant respectfully disagrees. At most Shen teaches the use of a store buffer and a load buffer to look for prior instructions contained in the buffers that alias with the current instruction being issued. *See Shen, pages 199-200.* Each time a particular instruction is encountered in a loop, the buffer is checked for aliasing instructions. That is, Shen has no memory of whether or not a particular instruction has been observed to previously alias with another instruction. Therefore, Shen does not teach previous detection of respective aliasings between read type instructions and write type instructions as required by the independent claims.

Thus, Shen does not disclose predicting aliasing between read type instructions and write type instructions based at least in part on respective displacements between the read type and write type instructions and on previous detection of respective aliasings between the read type instructions and the write type instructions as required by independent claim 1 and does not disclose the similar limitations as required by independent claims 16, 28, 38, 48, 54 and 62.

As such, for at least the reasons recited above, Shen is insufficient to anticipate independent claims 1, 16, 28, 38, 48, 54 and 62 and such indication is respectfully requested.

B. Dependent claims are not anticipated by Shen

Dependent claims 2-15, 17-27, 29-37, 39-47, 49-53, 55-57 and 63-66 depend upon and contain all the limitations of independent claims 1, 16, 28, 38, 48, 54 and 62, respectively. Accordingly, these dependent claims are themselves patentable over Shen under 35 U.S.C. § 102(a) for at least the reasons set forth above with regard to the independent claims.

II. THE REJECTION OF CLAIMS 16, 22, 29, and 38 UNDER 35 U.S.C. § 101 IS IMPROPER BECAUSE THE CLAIMS ARE DIRECTED TO STATUTORY SUBJECT MATTER

The Examiner rejected claims 16-47 under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter. Specifically, the Examiner alleges that claim 16 comprises steps of tracking and predicting, an abstract idea. *See final Office action, page 3.*

The Appellant respectfully asserts that independent claim 16 is directed toward statutory subject matter because one of the operations of the method is predicting a current instance of the read instruction will alias with a current instance of the write instruction when the displacement between the current instance of the read instruction and the current instance of the write instruction matches the displacement between previously aliased instances of the read instruction and write instruction. That is, the method tracks patterns of aliasing between particular read and write instructions to predict future aliasing, a tangible result. Further, independent claim 38 and dependent claims 22 and 29 produce tangible results in the form of bypassing data of the write instruction to the aliased read instruction with register information of the write instruction.

VIII. CONCLUSION

Appellant respectfully submits that all the appealed claims in this application are patentable and requests that the Board of Patent Appeals and Interferences direct allowance of the rejected claims.

Please charge Deposit Account No. 04-1415 in the amount of $500.00, for the filing fee of the Appeal Brief. The Appellant believes no further fees or petitions are required. However, if any such petitions or fees are necessary, please consider this a request therefor and authorization to charge Deposit Account No. 04-1415 accordingly.

Dated: Sept. 19, 2007

Respectfully submitted,

Gregory P. Durbin, Registration No. 42,503
Attorney for Appellant
USPTO Customer No. 66083

DORSEY & WHITNEY LLP
Republic Plaza Building, Suite 4700
370 Seventeenth Street
Denver, Colorado 80202-5647
Phone: (303) 629-3400
Fax: (303) 629-3450

IX. APPENDIX OF CLAIMS

1. A processor that predicts aliasing between read type instructions and write type instructions based at least in part on respective displacements between the read type and write type instructions and on previous detection of respective aliasings between the read type instructions and the write type instructions, and that bypasses data from the write type instructions to the corresponding predicted to alias read type instructions using register information of the aliasing write type instructions.

2. The processor of claim 1 wherein the data is bypassed if a threshold number of repeated aliases are detected.

3. The processor of claim 1 that includes encodings of read type instruction information, write type instruction information, and repeat aliasing.

4. The processor of claim 3 wherein the encodings comprise:

a read type instruction aliasing predictor table that indicates a static instruction identifier for a read type instruction, a displacement between the indicated read type instruction and a previously aliased write type instruction, and an alias prediction confidence indicator that indicates confidence of alias predictions;

a write type instruction aliasing predictor table that indicates the static instruction identifier for a write type instruction; and

an aliasing predictor management table that indicates a rename register identifier, an alias prediction counter that indicates a number of alias predictions, and a dynamic instruction identifier.

5. The processor of claim 4 wherein the static instruction identifier includes an address for the read or write-type instruction.

6. The processor of claim 4 wherein the dynamic instruction identifier monotonically increases with execution of a program that includes the instructions.

7. The processor of claim 4 that reduces the alias confidence prediction indicator for a read type instruction indicated in the read type instruction aliasing predictor table if the aliasing predictor management table does not indicate a write type instruction that corresponds to the read type instruction and the read type instruction's corresponding displacement.

8. The processor of claim 4 that reduces the alias confidence prediction indicator for a read type instruction indicated in the read type instruction aliasing predictor table if a misprediction of the read type instruction occurs.

9. The processor of claim 4 wherein the read type instruction aliasing predictor table includes a validity flag that indicates whether a threshold number of aliasings have been detected.

10. The processor of claim 1 wherein data bypasses comprise the processor substituting a register move instruction for the read type instruction.

11. The processor of claim 10, wherein a loadCheck instruction is inserted, which when executed by the processor, causes the processor to verify the predicted aliasing.

12. The processor of claim 10 wherein the register move instruction includes an integer-to-integer move instruction, a floating point-to-floating point move instruction, an integer-to-floating point move instruction, and a floating point-to-integer move instruction.

13. The processor of claim 1 wherein data bypasses comprise the processor mapping the read type instruction's destination register to the write type instruction's source register.

14. The processor of claim 13 that replaces the read type instruction with a loadCheck instruction, which when executed by the processor, causes the processor to verify the predicted aliasing.

15. The processor of claim 14 wherein the processor's verification of the predicted aliasing comprises interrogation of a data hazard detection module to ascertain whether addresses of the predicted to alias write type instruction and the read type instruction match, and verification of absence of intervening matching write type instructions.

16. A method comprising:

in a register rename stage, tracking a write type instruction and a read type instruction, instances of which have previously been indicated as aliased; and

predicting a current instance of the read type instruction will alias with a current instance of the write type instruction if displacement between the current instance of the read

type instruction and the current instance of the write type instruction matches displacement between previous aliased instances of the read type instruction and write type instruction.

17. The method of claim 16 wherein the displacement is measured with dynamic instruction identifiers, wherein the dynamic instruction identifiers identify corresponding instances of instructions with respect to program execution.

18. The method of claim 16 wherein the write type instruction and the read type instruction are tracked with their static identifier, wherein the static identifier identifies an instruction in a program and remains static during program execution.

19. The method of claim 18 wherein the static identifier includes an instruction address.

20. The method of claim 16 wherein a read type instruction includes a load instruction, a load halfword instruction, a load byte instruction, a load float instruction, a load double instruction, and a load multiple instruction.

21. The method of claim 16 wherein the write type instruction includes a store instruction, a store byte instruction, a store float instruction, a store double instruction, a store multiple instruction, and a store halfword instruction.

22. The method of claim 16 further comprising bypassing data of the write type instruction to the read type instruction with register information of the write type instruction.

23. The method of claim 22 wherein bypassing comprises mapping the read type instruction's destination register to the write type instruction's source register.

24. The method of claim 23 further comprising replacing the read type instruction with a loadCheck instruction, wherein the loadCheck instruction causes interrogation of a data hazard detection module to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match, and to ascertain whether there are any intervening matching write type instructions.

25. The method of claim 22 wherein bypassing comprises converting the read type instruction to a register move instruction.

26. The method of claim 25 further comprising inserting a loadCheck instruction, wherein the loadCheck instruction causes interrogation of a data hazard detection logic to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match, and to ascertain whether there are any intervening matching write type instructions.

27. The method of claim 16 embodied as a computer program product encoded in one or more machine-readable storage media.

28. A method comprising:

observing repeated aliasing between instances of a write type instruction and a read type instruction based at least in part on static identifiers of the instructions;

determining a displacement between the aliasing instances of the write type instruction and the read type instruction based on dynamic identifiers of the instruction instances;

predicting aliasing between a current instance of the read type instruction as identified by the static identifier thereof and a subsequent instance of the write type instruction identified with a dynamic identifier determined with a dynamic identifier of the current instance of the read type instruction and the displacement.

29. The method of claim 28 further comprising bypassing data of the subsequent instance of the write type instruction to the current instance of the read type instruction with register information of the subsequent instance of the write type instruction.

30. The method of claim 29 wherein bypassing the data comprises mapping the data destination of the current instance of the read type instruction to the data source of the subsequent instance of the write type instruction.

31. The method of claim 30 further comprising substituting a loadCheck instruction for the current instance of the read type instruction, wherein execution of the loadCheck instruction causes interrogation of a data hazard detection module to ascertain whether addresses of the current instance of the read type instruction and the subsequent instance of the write type instruction match.

32. The method of claim 31 wherein execution of the loadCheck instruction further causes verifying the absence of intervening matching write type instructions.

33. The method of claim 29 wherein bypassing the data comprises substituting a register move instruction for the read type instruction, wherein the move instruction moves data from the data source of the write type instruction to the data destination of the read type instruction.

34. The method of claim 28 wherein the dynamic identifiers monotonically increase with execution of a program that includes the instructions.

35. The method of claim 28 wherein the static identifiers include instruction addresses.

36. The method of claim 28 wherein the aliasing is predicted if the number of observed repeat aliasings exceeds a threshold.

37. The method of claim 28 embodied as a computer program product encoded in one or more machine-readable storage media.

38. A method comprising:

detecting aliasing between a first instance of a read type instruction and a first instance of a write type instruction;

determining displacement between the first instance of the read type instruction and the first instance of the write type instruction, wherein the displacement is with respect to program execution;

observing repeated aliasing between subsequent instances of the read type instruction and the write type instruction;

selecting a current instance of the write type instruction based at least in part on the displacement and a current instance of the read type instruction; and

bypassing data from a data source of the current instance of the write type instruction to a data destination of the current instance of the read type instruction.

39. The method of claim 38 further comprising verifying that the current instance of the write type instruction aliases with the current instance of the read type instruction.

40. The method of claim 38 wherein data bypass comprises changing the current instance of the read type instruction to a register move instruction.

41. The method of claim 40 further comprising inserting a loadCheck instruction, wherein the loadCheck instruction causes verification that the current instance of the read type instruction and the current instance of the write type instruction alias and verification of the absence of one or more intervening write type instructions.

42. The method of claim 38 wherein data bypasses comprise mapping the current instance of the read type instruction's architectural destination register to the current instance of the write type instruction's rename source register.

43. The method of claim 42 further comprising changing the current instance of the read type instruction to a loadCheck instruction, wherein the loadCheck instruction causes verification that the current instances of the read type instruction and the write type instruction alias and verification of the absence of one or more intervening write type instructions.

44. The method of claim 38 wherein the instances of the write type instruction have a same static identifier and different dynamic identifiers.

45. The method of claim 44 wherein the static identifiers include instruction addresses.

46. The method of claim 38 wherein the displacement is based at least in part on the dynamic identifiers of the instruction instances, wherein the dynamic identifiers monotonically increase with execution of a program that includes the instructions.

47. The method of claim 38 embodied as a computer program product encoded in one or more machine-readable storage media.

48. A computer program product encoded in one or more machine-readable storage media, the computer program product comprising:

a first sequence of instructions executable to, update a first encoding with a read type instruction's static identifier and a displacement between an instance of the read type instruction and an instance of a write type instruction observed as aliasing with the read type instruction instance if the read type instruction's static identifier is not indicated in the first encoding and to update the first encoding to indicate repeat aliasing if the read type instruction's static identifier is already indicated in the first encoding, update a second encoding with the write type instruction's static identifier;

a second sequence of instructions executable to update a third encoding with a dynamic identifier of an instance of a write type instruction if the static identifier thereof is indicated in the second encoding; and

a third sequence of instructions executable to bypass data from an instance of a write type instruction to an instance of a read type instruction with register information of the write type instruction instance based at least in part on displacement between the instances as indicated by corresponding dynamic identifiers.

49. The computer program product of claim 48 wherein data bypassing comprises the third sequence of instructions executable to map the read type instruction's destination register to the write type instruction's source register.

50. The computer program product of claim 49 wherein the read type instruction's destination register includes an architectural register and the write type instruction's source register includes a rename register.

51. The computer program product of claim 50 wherein the third sequence of instructions are further executable to re' lace the read type instruction instance with a loadCheck instruction, which when executed causes verification that the read type instruction and the write type instruction alias to the same address and verification of the absence of one or more intervening write type instructions aliasing to the same address.

52. (Previously Presented) The computer program product of claim 48 wherein bypassing data comprises the third sequence of instructions executable to replace the read type instruction with a register move instruction.

53. The computer program product of claim 52, further comprising the third sequence of instructions executable to insert a loadCheck instruction proximate with the register move instruction, wherein execution of the loadCheck instruction causes verification that the read type instruction and the write type instruction alias to the same address and verification of the absence of one or more intervening write type instructions aliasing to the same address.

54. An apparatus comprising:

a data hazard detection module; and

means for predicting aliasing between a current instance of a read type instruction and a current instance of a write type instruction based on displacement between the current

instance of the instructions and displacement between previously observed aliased instances of the read type instruction and a the write type instruction.

55. The apparatus of claim 54 further comprising means for bypassing data from the write type instruction to the read type instruction with register information of the write type instruction.

56. The apparatus of claim 54 wherein the read type instruction includes a load instruction, a load halfword instruction, a load byte instruction, a load float instruction, a load double instruction, and a load multiple instruction.

57. The apparatus of claim 54 wherein the write type instruction includes a store instruction, a store byte instruction, a store float instruction, a store double instruction, a store multiple instruction, and a store halfword instruction.

58. An apparatus comprising:

a data hazard detection module; and

rename unit coupled with the data hazard detection module, the rename unit to rename registers of instructions and to predict aliasing between instances of read type instructions and instances of write type instructions based at least in part on respective displacements between the instruction instances, wherein the rename unit includes one or more structures operable to, track read type instructions indicated by the data hazard detection module as aliasing and track repeat aliasing of the tracked read type instructions, and to indicate displacements between instances of the tracked read type instructions and aliased instances of the write type instructions;

indicate write type instructions indicated by the data hazard detection module as aliasing; and

indicate instances of the write type instructions encountered in the rename unit that are indicated in the second structure.

59. The apparatus of claim 58 wherein the data hazard detection module includes a memory disambiguation buffer or a load/store.

60. The apparatus of claim 58 further comprising an instruction scheduling unit coupled with the rename unit and the data hazard detection module.

61. The apparatus of claim 58 wherein the structures include hardware tables and logical structures instantiable in memory.

62. An apparatus comprising:

an alias predictor, including one or more structures to host indications of write type instructions and particular instances of the write type instructions and read type instructions, respective execution displacements between particular instances of read and write type instructions, and register information of the particular write type instruction instances, the alias predictor operable to predict aliasings between read and write type instruction instances based, at least in part, on the structures and indications of detected aliasings between the instruction instances;

a rename unit coupled with the alias predictor, the rename unit to supply register information for write type instruction instances to the alias predictor; and

a data hazard detection unit coupled with the alias predictor, the data hazard detection unit to detect aliasing between particular instances of read and write type instructions and to indicate detected aliasings to the alias predictor.

63. The apparatus of claim 62, wherein the indications of particular instances of instructions include instruction instance addresses.

64. The apparatus of claim 63, wherein the instruction instances addresses include one or more of static identifiers and dynamic identifiers.

65. The apparatus of claim 62, wherein the structures comprise:

a first structure operable to indicate read type instructions with static identifiers thereof, respective execution displacements with potentially aliasing instances of write type instructions, and respective alias prediction confidence;

a second structure operable to indicate write type instructions with static identifiers thereof; and

a third structure operable to indicate particular instances of write type instructions with dynamic identifiers and register information thereof.

66. The apparatus of claim 65 further comprising:

the first and second structures operable to also indicate alias prediction validity; and the third structure operable to also indicate pending unverified alias predictions.

X. EVIDENCE APPENDIX

None.

XI. RELATED PROCEEDINGS APPENDIX

None.